1。它们是单目运算符，只有一个操作数。

“++”出现在变量前面和后面是有区别的：“++”在变量前面时，先进行自增操作，再执行本行语句；“++”在变量后面时，待本行语句执行完成后，再进行自增操作。自减运算符出现在变量前后的运算步骤与自增运算符类似。当自增或自减完成后，新值会替换旧值。

请注意自增和自减运算符只能针对变量，不能用于表达式和数值。

延伸阅读

[1] KERNIHAN B W，RITCHIE D M. C 程序设计语言[M]. 徐宝文，李志，译. 2 版. 北京：机械工业出版社，2004：37.

[2] BJARNE S. C++程序设计语言(第 1~3 部分)[M]. 王刚，杨巨峰，译. 4 版. 北京：机械工业出版社，2016：238-239.

（金靖）

1.2.4.5 三目运算

三目运算符，又称为条件运算符，是 C++中唯一具有 3 个操作数的运算符。

`a ? b : c` 表示对表达式 a 求值，若值为真则执行表达式 b，否则执行表达式 c。

参考词条

if 语句、switch 语句、多层条件语句

延伸阅读

KERNIHAN B W，RITCHIE D M. C 程序设计语言[M]. 徐宝文，李志，译. 北京：机械工业出版社，2004：184.

（金靖）

1.2.4.6 位运算：与(&)、或(|)、非(~)、异或(^)、左移(<<)、右移(>>)

位运算直接对整数在内存中的二进制位进行运算。常用位运算包括与、或、非、异或、左移、右移等。

1. 与(&)

将参与运算的两个操作数各自对应的二进制位进行“逻辑与”操作。只有对应的两个二进制位均为 1 时，对应二进制位的运算结果才为 1，否则为 0。

2. 或(|)

将参与运算的两个操作数各自对应的二进制位进行“逻辑或”操作。只有对应的两个二进制位均为 0 时，对应二进制位的运算结果才是 0，否则为 1。

3. 非(~)

单目运算符，它将操作数中的二进制位 0 变成 1，把 1 变成 0。

4. 异或(^)

将参与运算的两个操作数各自对应的二进制位进行“异或”操作。当对应的两个二进制位不相同时，对应二进制位的运算结果为1，当对应的两个二进制位相同时，运算结果为0。

5. 左移(<<)

左移运算是将左操作数的二进制形式全部向左移动若干位(右操作数的值)后得到的值。左移时，高位丢弃，右边低位补0。左移 n 位，就是乘以 2^n。

6. 右移(>>)

右移运算的操作方式和左移运算类似，不同的是移动方向，移出最右边的位被丢弃。同样，右移 n 位相当于左操作数除以 2^n，并且将结果向下取整。对于无符号数，右移时高位总是补0。对于有符号数，在右移时，符号位(即最高位)一起移动。如果原符号位为1(代表负数)，则右移时右边高位都补1；如果原符号位为0(代表正数)，则右移时高位补0，即保持原有的正负。

参考词条

1. 进制与进制转换：二进制、八进制、十进制、十六进制
2. 状态压缩动态规划

延伸阅读

[1] KERNIHAN B W，RITCHIE D M. C程序设计语言[M]. 徐宝文，李志，译. 2版. 北京：机械工业出版社，2004：38-39.

[2] BJARNE S. C++程序设计语言(第1~3部分)[M]. 王刚，杨巨峰，译. 4版. 北京：机械工业出版社，2016：236-238.

(金靖)

1.2.5 数学库常用函数

数学库常用函数包括绝对值函数、四舍五入函数、上取整函数、下取整函数、平方根函数、常用三角函数、对数函数、指数函数等，如表1.1所示。

表1.1 数学库常用函数

函数	语法	功能
绝对值函数	abs(x)	返回 x 的绝对值
四舍五入函数	round(x)	返回离 x 最近的整数值
上取整函数	ceil(x)	返回大于或等于 x 的最小的整数值
下取整函数	floor(x)	返回小于或等于 x 的最大的整数值
正弦函数	sin(x)	返回三角函数的正弦值，x 是弧度

（续）

函数	语法	功能
余弦函数	cos(x)	返回三角函数的余弦值，x 是弧度
正切函数	tan(x)	返回三角函数的正切值，x 是弧度
反正弦函数	asin(x)	返回三角函数的反正弦值，x 是弧度
反余弦函数	acos(x)	返回三角函数的反余弦值，x 是弧度
反正切函数	atan(x)	返回三角函数的反正切值，x 是弧度
对数函数	log(x)	返回 x 的自然对数
指数函数	pow(x,y)	返回 x^y 的值
平方根函数	sqrt(x)	返回$\sqrt{x}$的值

对于绝对值函数，需注意，在 stdlib. h 中定义的是 C 语言的绝对值函数 int abs(int x)，使用 double fabs(double x)可以计算实数型的绝对值。在 cmath 头文件中定义的是 C++ 版本的绝对值函数 double abs(double x)，可以支持整型和实数型的绝对值计算。

延伸阅读

KERNIHAN B W，RITCHIE D M. C 程序设计语言[M]. 徐宝文，李志，译. 2 版. 北京：机械工业出版社，2004：228-229.

（金靖）

1.2.6 结构化程序设计

1.2.6.1 顺序结构、分支结构和循环结构

结构化程序设计是程序设计的一种原则，该原则以模块化为中心，将复杂的计算任务划分为许多子任务，并针对每个子任务(模块或函数)进行设计。

结构化程序设计遵循“程序=算法+数据结构”的原则，将算法和数据结构分别进行独立设计，其中以算法的设计为主。在保证程序仅存在一个入口、一个出口的基础上，采用自顶向下、逐步求精的设计方法，利用顺序、分支、循环这三个基本控制结构连接各模块，最终构成整体。

1. 顺序结构

顺序结构是最简单的线性结构，其逻辑是自上而下，依次执行各语句。顺序结构可独立构成程序，但更常见的是作为分支、循环结构的组成部分，与其他结构共同构成更加复杂的逻辑。变量定义、赋值和函数调用等皆可作为顺序结构的组成部分。

2. 分支结构

分支结构依据判断条件的真假对部分模块进行选择性执行，而非严格依照语句出现的前后顺序。常见的分支语句有单分支(if…)、双分支(if…else…)、嵌套分支(if…else if…else…)和 switch 语句。

3. 循环结构

循环结构依据判断条件的真假对部分语句进行循环执行，直至不满足判断条件后终止循环。依据判断与循环体的先后顺序分为“当型”循环和“直到型”循环。“当型”循环先判断条件，若满足条件则执行循环体，常见语句有 for 循环，while 循环；“直到型”循环先执行循环体，执行完成后判断条件，若不满足条件则跳出循环，常见语句有 do…while 循环。

参考词条

1. if 语句、switch 语句、多层条件语句
2. for 语句、while 语句、do while 语句
3. 多层循环语句

（金靖）

1.2.6.2 自顶向下、逐步求精的模块化程序设计

结构化程序设计具有自顶向下、逐步求精和模块化三个特点。

自顶向下，即进行程序设计时，应先考虑整体，后考虑细节，从最上层的总目标开始设计，并逐步将问题细化。逐步求精，即面对复杂的问题时，应设计一些子目标作为过渡，逐步细化。因此，程序设计会将总目标细分为许多子任务，其中每个子任务成为一个模块——此即结构化程序设计中的模块化特点。

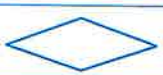
（金靖）

1.2.6.3 程序流程图的概念及流程图描述

程序流程图，又称程序框图，即用统一规定的标准符号描述程序运行具体步骤的图形表示，是程序设计最根本的依据。

程序流程图的绘制采用简单规范的符号，因此具有绘制简单、结构清晰、易于描述和理解的特点，在各个领域受到广泛使用。具体的符号与控制结构的图例如表 1.2 所示。

表 1.2 程序流程图符号及控制结构图例

符号	名称	功能
	起止框	表示流程开始或结束
	输入输出框	表示输入输出功能
	处理框	表示一般的处理功能
	判断框	表示对一个给定的条件进行判断，根据给定的条件是否成立决定如何执行其后的操作
	流程线	表示流程的路径和方向

参考词条

算法描述：自然语言描述、流程图描述、伪代码描述

（金靖）

1.2.7 数组

1.2.7.1 数组与数组下标

数组是一种数据结构。它可以存储一个固定长度的、由同类型元素构成的序列。数组由连续的内存位置组成。数组中每个元素都有其对应的下标。

数组支持随机访问。随机访问指的是，对于任意一个下标，能在任意时刻访问数组中该下标对应的元素。

在 C++中，数组的常用声明方法为：

```
T a[N]; // T 代表数据类型,a 代表数组名,N 代表数组长度
```

数组长度即数组包含的元素个数，N 必须是常数或常量表达式，并且其值必须是正整数。这条语句相当于声明了 $a[0],a[1],\cdots,a[N-1]$ 这 N 个元素，方括号内的数为该元素的下标，在程序中调用 $a[i]$ 即可访问下标为 i 的元素。

对于下标 i，如果它不在 0 到 $N-1$ 的范围内，则访问 $a[i]$ 会造成数组越界。这是一种未定义行为（undefined behavior），会导致不可预测的后果，例如经常会造成运行时错误（runtime error）。因此在程序设计中，应避免出现这种错误。

需要特别注意的是，在 C++中数组的下标从 0 开始（即 0-indexed），因此若声明大小为 N 的数组，访问 $a[N]$ 就会造成数组越界。为了防止这类越界情况的发生，可以声明比所需稍大一点的数组。

如果直接声明，全局数组会被默认初始化（基础数据类型被赋值为 0，结构体则调用默认构造函数），而局部数组不会。往往声明局部数组时要用大括号初始化。

```
int a[3] = { 1,2,0 };
// 声明长度为 3 的局部数组 a,其初值为 a[0]=1,a[1]=2,a[2]=0;
// 等价写法 1:int a[] = { 1,2,0 }; (省略数组长度,按照初始化列表的长度,数组长度为 3)
// 等价写法 2:int a[3] = { 1,2 }; (省略数组末尾的 0)
int b[3] = {};
// 声明长度为 3 的局部数组 b,其初值为 b[0]=0,b[1]=0,b[2]=0;
```

在 C++中，用如下语句可以定义一个二维数组：

```
T a[N][M];
```